**КОНТРОЛЬНАЯ РАБОТА**

Решение задачи оптимизации методом генетического алгоритма

## **Введение**

алгоритм переменная кроссинговер

В процессе выполнения данной контрольной работе была написана программа Matlab, основной задачей которой является проведение условной минимизации заданной функции нескольких переменных на основе применения генетического алгоритма (ГА). Генетический алгоритм был реализован по уравнениям и данным, заданным согласно варианту. Также была реализована настройка ГА встроенной функцией gaoptimset. Результат был сравнен с результатом написанной программы. Также были проведены исследования работы ГА с необходимыми графическими иллюстрациями в соответствии с вариантом.

## **1. Постановка задачи**

Цель работы.

Научиться реализовывать условную минимизацию заданной функции нескольких переменных на основе применения генетического алгоритма (ГА) в среде Matlab, научиться реализовывать алгоритм, не используя окно тулбокса, настраивать ГА встроенной функцией MATLAB gaoptimset и исследовать работу ГА, реализуя различные графики, диаграммы, таблицы.

Задание на контрольную работу.

) Провести условную минимизацию заданной функции нескольких переменных на основе применения генетического алгоритма (ГА), программно реализованного в Matlab (использовать только стандартную функцию).

) Составить программу решения задачи в Matlab в виде m-файла, не используя окно тулбокса.

) Для настройки ГА использовать функцию gaoptimset. Все задаваемые опции прописать в разработанной программе явным образом.

) Провести исследования работы ГА с необходимыми графическими иллюстрациями в соответствии с вариантом.

Индивидуальное задание.

Вариант №24.

5) Дана следующая функция:



1) Построить график заданной функции при *n* = 2. Определить визуально, имеет ли данная функция глобальный минимум.

) Провести оптимизацию заданной функции в Matlab (с помощью генетического алгоритма): найти глобальный минимум.

3) Для решения задачи составить программу в среде Matlab (m-файл).

4) Результат определить как среднее по 20 решениям.

5) В отчете отразить ход решения задачи:

- представить график средних и наилучших по поколениям значений целевой функции;

провести исследование зависимости решения от вида функции отбора родителей для кроссинговера и мутации потомков.

## **2. Теоретические сведения**

### **2.1 Понятие генетического алгоритма**

Генетические алгоритмы - это адаптивные методы поиска, которые в последнее время используются для решения задач оптимизации. В них используются как аналог механизма генетического наследования, так и аналог естественного отбора. При этом сохраняется биологическая терминология в упрощенном виде и основные понятия линейной алгебры. Основной идеей генетических алгоритмов является организация «борьбы за существование» и «естественного отбора» среди этих пробных решений.

Рассмотрим принципы работы генетических алгоритмов на максимально простом примере. Пусть требуется найти глобальный минимум функции на отрезке [0; 7]. На этом отрезке функция принимает минимальное значение в точке x = 1. Очевидно, что в точке x = 6 функция попадает в локальный минимум. Если для нахождения глобального минимума использовать градиентные методы, то в зависимости от начального приближения можно попасть в данный локальный минимум.

Рассмотрим на примере данной задачи принцип работы генетических алгоритмов. Для простоты положим, что x принимает лишь целые значения, т.е. x ∈ {0, 1, 2, 3, 4, 5, 6, 7}. Это предположение существенно упростит изложение, сохранив все основные особенности работы генетического алгоритма. Выберем случайным образом несколько чисел на отрезке [0; 7]: {2, 3, 5, 4}. Запишем пробные решения в двоичной форме: {010, 011, 101, 100}. Поскольку генетические алгоритмы используют биологические аналогии, то и применяющаяся терминология напоминает биологическую. Так, одно пробное решение, записанное в двоичной форме, мы будем называть особью или хромосомой, а набор всех пробных решений - популяцией. Как известно, принцип естественного отбора заключается в том, что в конкурентной борьбе выживает наиболее приспособленный. В нашем случае приспособленность особи определяется целевой функцией: чем меньше значение целевой функции, тем более приспособленной является особь, т.е. пробное решение, использовавшееся в качестве аргумента целевой функции.

Теперь приступим к процессу размножения: попробуем на основе исходной популяции создать новую, так чтобы пробные решения в новой популяции были бы ближе к искомому глобальному минимуму целевой функции. Для этого сформируем из исходной популяции брачные пары для скрещивания. Поставим в соответствие каждой особи исходной популяции случайное целое число из диапазона от 1 до 4. Будем рассматривать эти числа как номера членов популяции. При таком выборе какие-то из членов популяции не будут участвовать в процессе размножения, так как образуют пару сами с собой. Какие-то члены популяции примут участие в процессе размножения неоднократно с различными особями популяции. Процесс размножения (рекомбинация) заключается в обмене участками хромосом между родителями. Например, пусть скрещиваются две хромосомы 111111 и 000000. Определяем случайным образом точку разрыва хромосомы, пусть, это будет 3: 111|111 000|000. Теперь хромосомы обмениваются частями, стоящими после точки разрыва, и образуют двух новых потомков: 111000 и 000111.

Следующим шагом в работе генетического алгоритма являются мутации, т.е. случайные изменения полученных в результате скрещивания хромосом. Пусть вероятность мутации равна 0,3. Для каждого потомка возьмем случайное число на отрезке [0; 1], и если это число меньше 0,3, то инвертируем случайно выбранный ген (заменим 0 на 1 или наоборот).

Как видно на примере, мутации способны улучшить (первый потомок) или ухудшить (четвертый потомок) приспособленность особи-потомка. В результате скрещивания хромосомы обмениваются «хвостами», т.е. младшими разрядами в двоичном представлении числа. В результате мутаций изменению может подвергнуться любой разряд, в том числе, старший. Таким образом, если скрещивание приводит к относительно небольшим изменениям пробных решений, то мутации могут привести к существенным изменениям значений пробных решений.

Теперь из четырех особей-родителей и четырех полученных особей потомков необходимо сформировать новую популяцию. В новую популяцию отберем четыре наиболее приспособленных особей из числа «старых» особей и особей-потомков.

В результате получим новое поколение. Получившуюся популяцию можно будет вновь подвергнуть кроссинговеру, мутации и отбору особей в новое поколение. Таким образом, через несколько поколений мы получим популяцию из похожих и наиболее приспособленных особей. Значение приспособленности наиболее «хорошей» особи (или средняя приспособленность по популяции) и будет являться решением нашей задачи. Следуя этому, в данном случае, взяв наиболее приспособленную особь 001 во втором поколении, можно сказать, что минимумом целевой функции является значение −5,42, соответствующее аргументу x = 1. Тем самым попадания в локальный минимум удалось избежать! На данном примере разобран вариант простого генетического алгоритма. При дальнейшем использования ГА к разным задачам возможно моделирование основных операторов алгоритма.

### **.2 Понятие минимизации функции многих переменных**

Градиентом дифференцируемой функции f(x) в точке х[0] называется n-мерный вектор f (x[0]), компоненты которого являются частными производными функции f(х), вычисленными в точке х[0], т.е. f' (x[0]) = (дf (х[0])/дх1, …, дf (х[0])/дхn)T.

Этот вектор перпендикулярен к плоскости, проведенной через точку х[0], и касательной к поверхности уровня функции f(x), проходящей через точку х[0].В каждой точке такой поверхности функция f(x) принимает одинаковое значение. Приравнивая функцию различным постоянным величинам С0, С1,…, получим серию поверхностей, характеризующих ее топологию.

Вектор-градиент направлен в сторону наискорейшего возрастания функции в данной точке. Вектор, противоположный градиенту (-f’ (х[0])), называется антиградиентом и направлен в сторону наискорейшего убывания функции. В точке минимума градиент функции равен нулю. На свойствах градиента основаны методы первого порядка, называемые также градиентным и методами минимизации. Использование этих методов в общем случае позволяет определить точку локального минимума функции.

Очевидно, что если нет дополнительной информации, то из начальной точки х[0] разумно перейти в точку х, лежащую в направлении антиградиента - наискорейшего убывания функции. Выбирая в качестве направления спуска р[k] антиградиент - f’ (х[k]) в точке х[k], получаем итерационный процесс вида х [k+1] = x[k] - akf' (x[k]), аk > 0; k=0, 1,…

В координатной форме этот процесс записывается следующим образом:

xi[k+1]=хi[k] - akf (x[k]) /xi; i = 1,…, n; k= 0, 1, 2,…

В качестве критерия останова итерационного процесса используют либо выполнение условия малости приращения аргумента || x [k+l] - x[k] || <= e, либо выполнение условия малости градиента|| f’ (x[k+l]) || <= g. Здесь e и g - заданные малые величины.

Возможен и комбинированный критерий, состоящий в одновременном выполнении указанных условий. Градиентные методы отличаются друг от друга способами выбора величины шага аk. При методе с постоянным шагом для всех итераций выбирается некоторая постоянная величина шага. Достаточно малый шаг аk обеспечит убывание функции, т.е. выполнение неравенства f (х[k+1]) = f (x[k] - akf’ (x[k])) < f (x[k]).

Однако это может привести к необходимости проводить неприемлемо большое количество итераций для достижения точки минимума. С другой стороны, слишком большой шаг может вызвать неожиданный рост функции либо привести к колебаниям около точки минимума (зацикливанию). Из-за сложности получения необходимой информации для выбора величины шага методы с постоянным шагом применяются на практике редко. Более экономичны в смысле количества итераций и надежности градиентные методы с переменным шагом, когда в зависимости от результатов вычислений величина шага некоторым образом меняется.

## **3. Решение задания**

### **3.1 Построение графика функции**

График заданной функции представлен на рисунке 3.1.



Рис. 3.1 - График заданной функции

### **3.2 Оптимизация заданной функции**

В процессе оптимизации целевой функции в командном окне Matlab отображается информация о следующих параметрах (рис. 3.2):

номере поколения и количестве вычислений значений целевой функции;

наилучших значениях целевой функции;

среднем значении целевой функции;

количестве вычислений в рамках одного поколения;

причины завершения расчета.



Рисунок 3.2 - Текущая информация о параметрах

### **.3 Отражение хода решения задачи**

В результате решения задачи оптимизации заданной функции искомые *xi* стремятся к значениям, показанным на рис. 3.2.

Результаты решения (значения целевой функции) представлены в таблице 3.1.

|  |  |
| --- | --- |
|  | *n*=2 |
| -27 |  |

В процессе решения выводится график средних и наилучших по поколениям значений целевой функции (рис. 3.3).



Рис. 3.3 - График средних и наилучших по поколениям значений целевой функции

### **.4 Исследование зависимости решения от вида функции отбора родителей для кроссинговера и мутации потомков**

Для проведения исследования задавались различным видом функции отбора родителей для кроссинговера и мутации потомков:

1) @selectionremainder;

) @selectionuniform;

) @selectionstochunif;

) @selectionroulette;

) @selectiontournament.

Результаты расчетов представлены на рисунке 3.4.



Рис. 3.4 - График зависимости решения от вида функции отбора родителей для кроссинговера и мутации потомков

Как видно из графика, значение целевой функции с изменением вида функции отбора родителей для кроссинговера и мутации потомков увеличивается.

## **Вывод**

В процессе выполнения данной контрольной работы была реализована программа Matlab, основной задачей которой является проведение условной минимизации заданной функции нескольких переменных на основе применения генетического алгоритма (ГА). Генетический алгоритм был реализован по уравнениям и данным, заданным согласно варианту. Также была реализована настройка ГА встроенной функцией gaoptimset. Результат был сравнен с результатом написанной программы. Также были проведены исследования работы ГА с необходимыми графическими иллюстрациями в соответствии с вариантом.

## **Список литературы**

1) Акулич И.Л. Математическое программирование в примерах и задачах. - М.: Изд. «Высшая школа», 1986.

2) Гершкович Ю.Б. Методы оптимизации и их применение для управления процессами в нефтяной промышленности. - М.: МИНГ им. И.М. Губкина, 1988.

) Моисеев Н.Н., Иванилов Ю.П. Методы оптимизации. - М.: Наука, 1978.

) Новгородцев А. Расчет электрических цепей в «MATLAB». - Спб.: Питер, 2004.

## **Приложение**

## **Текст программы**

Файл a2.m

= 20;= 2;% число переменных x

(1) = -30;(1) = 30;(2) = -10;(2) = 10;

% построение графика функции 2-х переменных

if nvars == 2= (Xmax(1) - Xmin(1))/49;= (Xmax(2) - Xmin(2))/49;= Xmin(1):dx1: Xmax(1);= Xmin(2):dx2: Xmax(2);= length(x1);= length(x2);i=1:1:n1j=1:1:n2(i, j) = 2\*x1 (i)^2-4\*x1 (i)^4+1/2\*x1 (i)^6+x1 (i)^2\*x2 (j)^2;(x1, x2, Y), grid on;('x1');('x2');('f(x)');

i1=1:1:5i11,= @selectionremainder;2,= @selectionuniform;3,= @selectionstochunif;4,= @selectionroulette;5,= @selectiontournament;= [Xmin; Xmax];= gaoptimset;= gaoptimset (options, 'PopInitRange', PIR);= gaoptimset (options, 'PopInitRange', []);= gaoptimset (options, 'PopulationSize', 100);= gaoptimset (options, 'EliteCount', 5);= gaoptimset (options, 'CrossoverFraction', 0.8);= gaoptimset (options, 'ParetoFraction', 0.05);= gaoptimset (options, 'MigrationDirection', 'both');= gaoptimset (options, 'MigrationInterval', 1);= gaoptimset (options, 'MigrationFraction', 1);= gaoptimset (options, 'Generations', 100);= gaoptimset (options, 'TimeLimit', 400);= gaoptimset (options, 'FitnessLimit', - Inf);= gaoptimset (options, 'StallGenLimit', 50);= gaoptimset (options, 'StallTimeLimit', 600);= gaoptimset (options, 'TolFun', 1e-6);= gaoptimset (options, 'TolCon', 1e-6);= gaoptimset (options, 'InitialPopulation', []);= gaoptimset (options, 'InitialScores', []);= gaoptimset (options, 'InitialPenalty', []);= gaoptimset (options, 'PenaltyFactor', []);= gaoptimset (options, 'CreationFcn', @gacreationuniform);= gaoptimset (options, 'FitnessScalingFcn', @fitscalingrank);= gaoptimset (options, 'SelectionFcn', SF);= gaoptimset (options, 'CrossoverFcn', {@crossoverheuristic, 1.2});= gaoptimset (options, 'MutationFcn', @mutationadaptfeasible);= gaoptimset (options, 'DistanceMeasureFcn', []);= gaoptimset (options, 'HybridFcn', []);= gaoptimset (options, 'Display', 'iter');= gaoptimset (options, 'PlotFcns', {@gaplotbestf});= gaoptimset (options, 'PlotInterval', 1);

k=1:1:N

[X (k,:), FVal(k)]=ga (@ex95, nvars, [], [], [], [], Xmin, Xmax, [], options);(i1) = mean(FVal),j=1:1:nvars\_(i1, j) = mean (X(:, j));\_(i1,:),

(3);(1:1:i1, F), grid on;

Файл ex95.m

y = ex95 (x)(1)^2-12\*x(1)+11+10\*cos (pi/2\*x(1))+8\*sin (5\*pi\*x(1)) - 1/sqrt(5)\*exp (-1\*((x(1) - 0.5)^2)/2);

end